

آکادمی رباتک

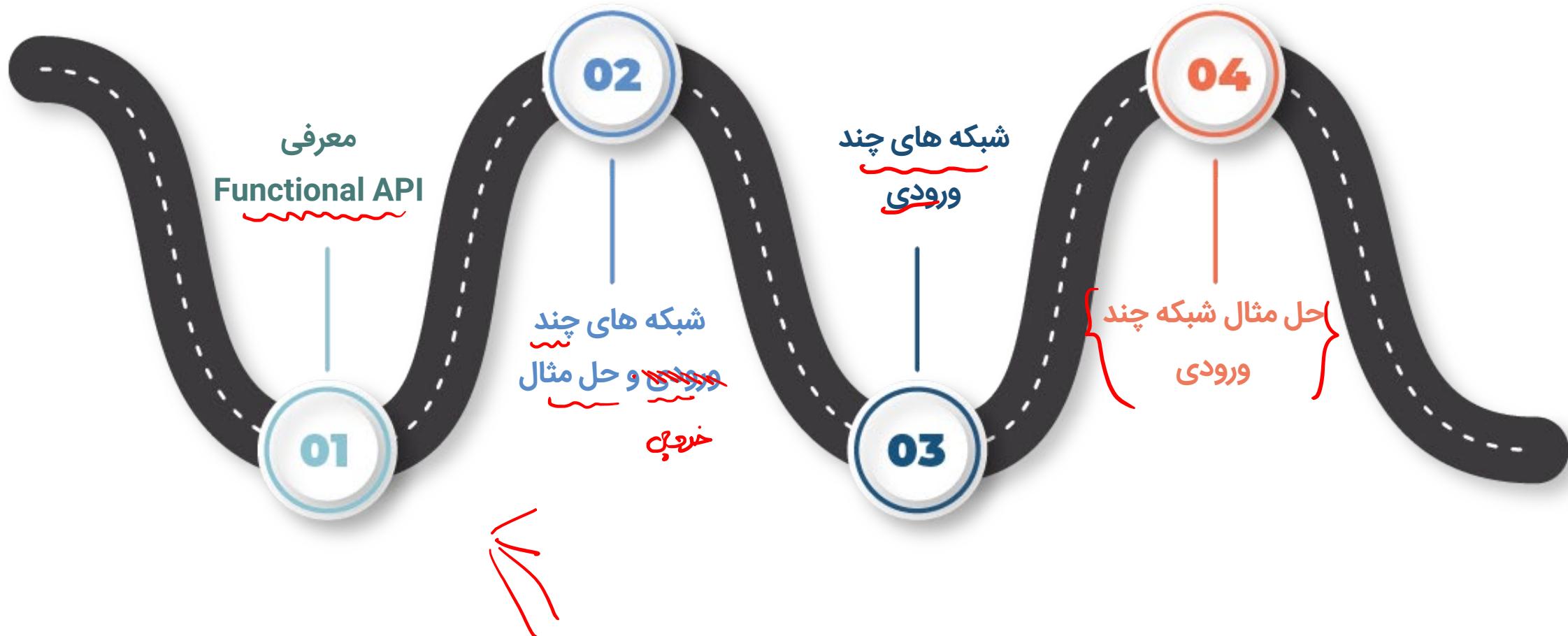
دوره تنسورفلو پیشرفته

جلسه دوم : شبکه های چند ورودی و چند خروجی

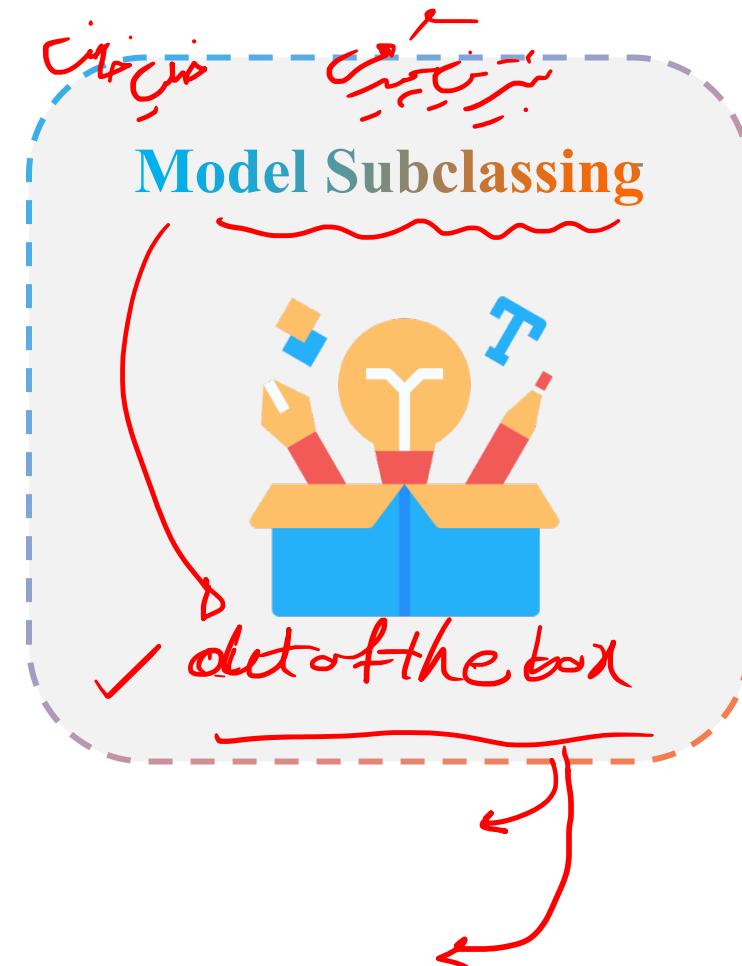
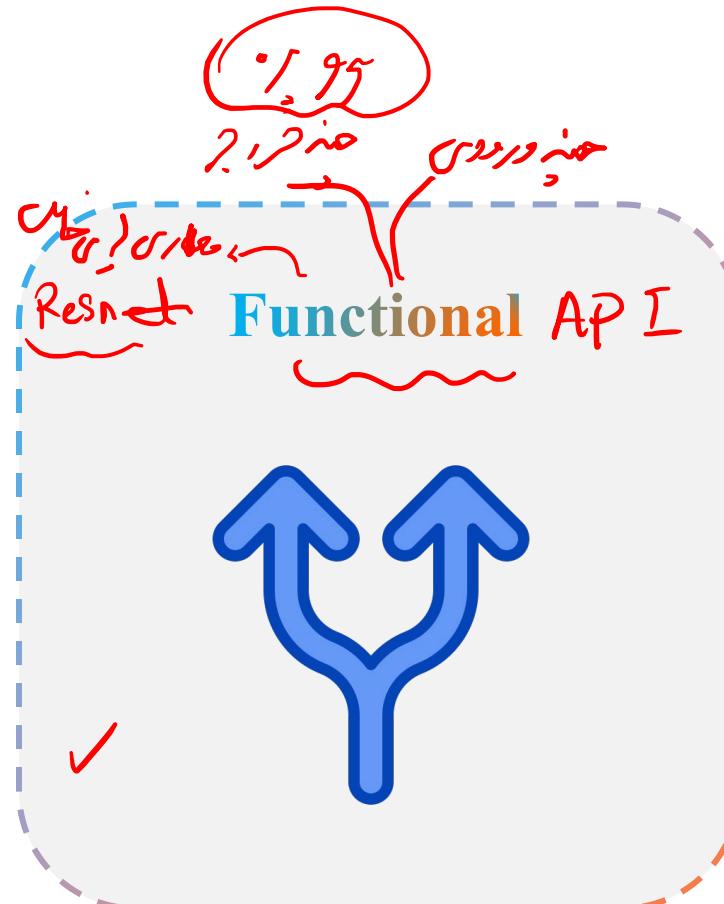
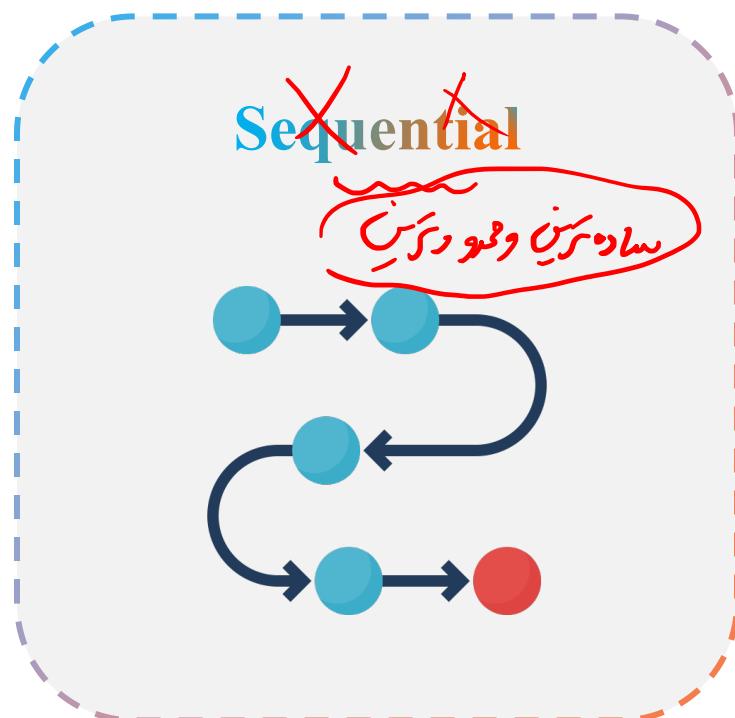
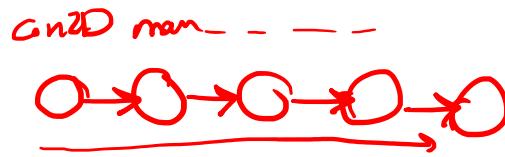


آنچه امروز خواهیم گفت :

شبکه های چند گره ای
و روش های آنها

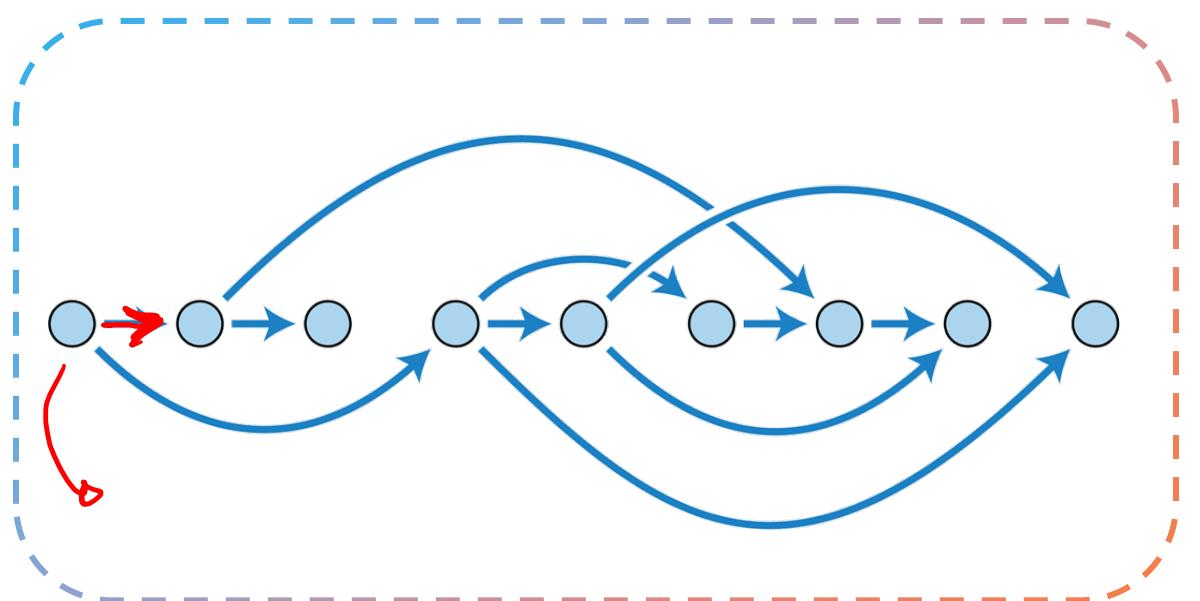


سه روش پیاده سازی شبکه در Keras



گرافیک متریک (DAG)
جربه های

Functional API



Node :

Connection :

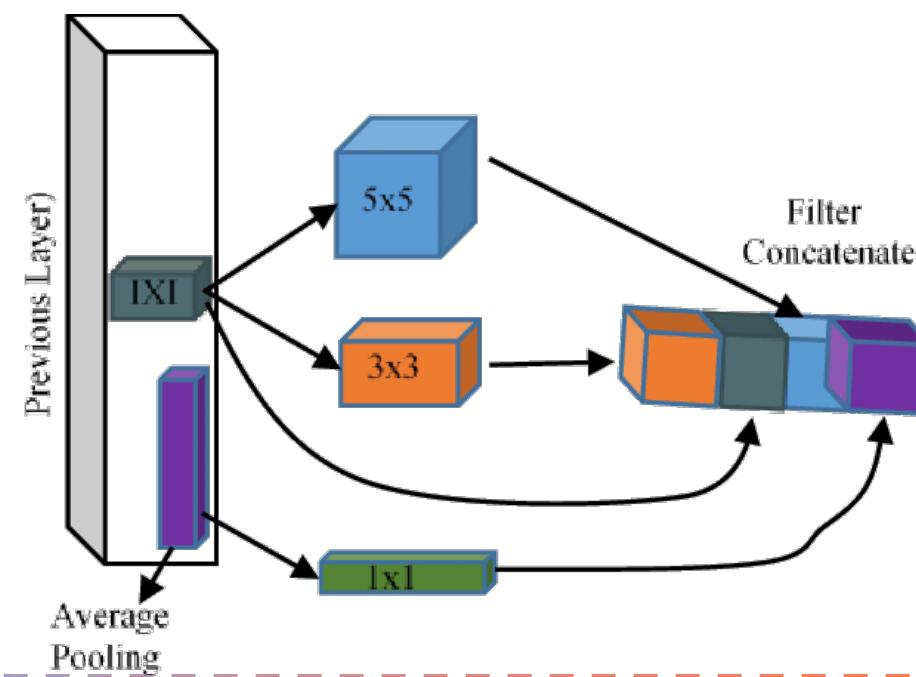
$\lambda \approx \delta \rightarrow \text{Conv2D}$
 Maxpool2D

کاربرد API Functional

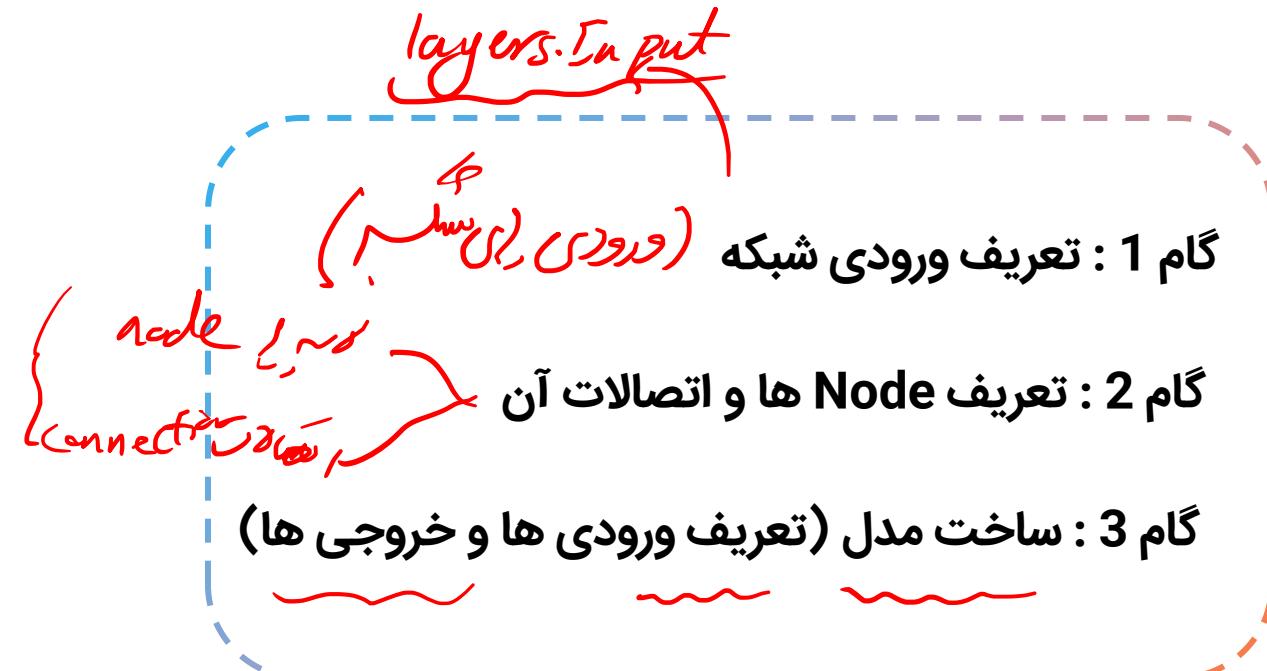
model subclass

ساختارهای موازی مثل ResNet و Inception

ساختارهای چند ورودی و یا چند خروجی



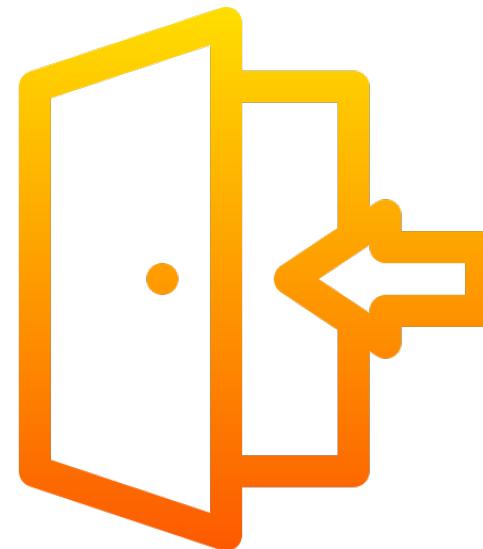
نحوه ایجاد یک مدل Functional API



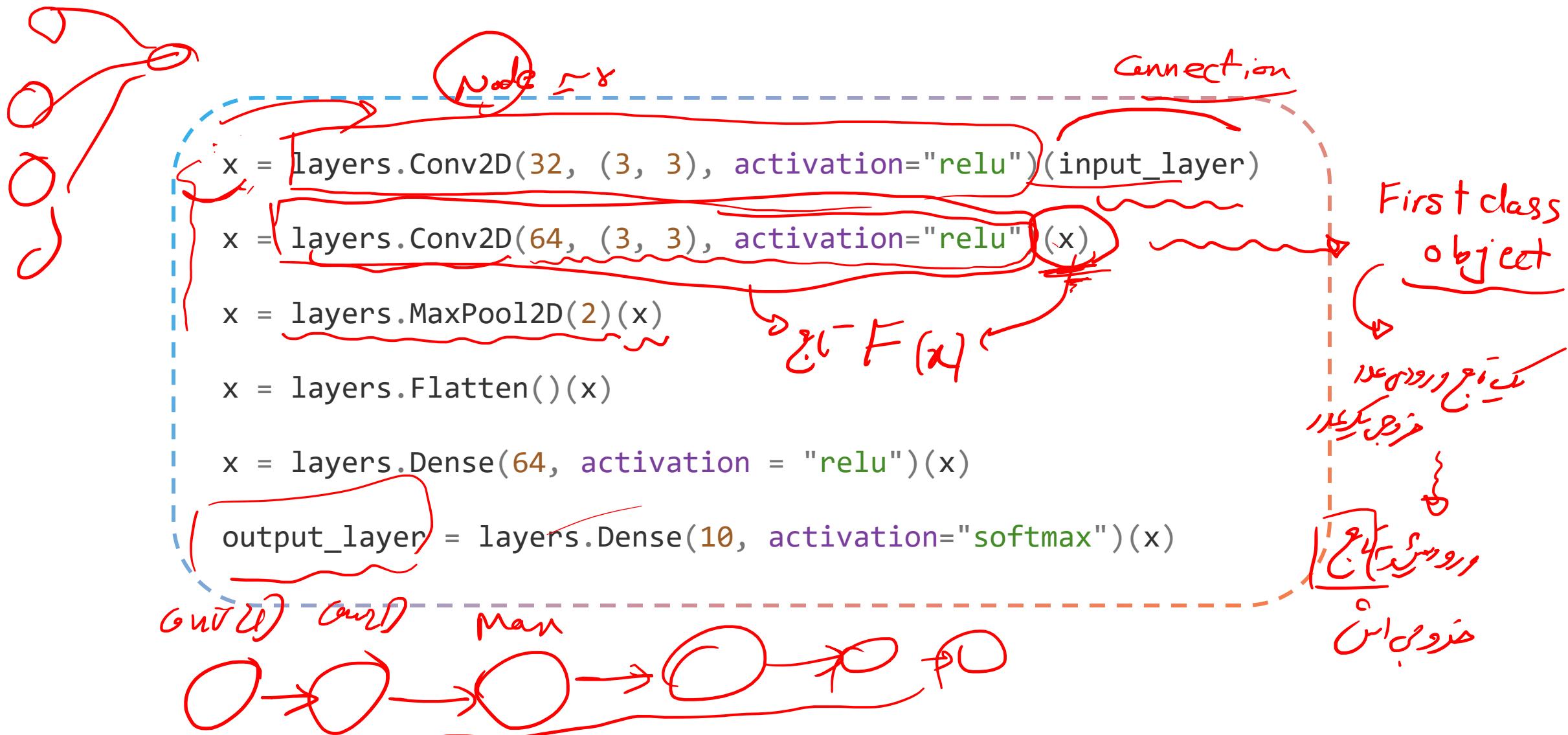
گام 1 : تعریف لایه ورودی

```
input_layer = layers.Input(shape=(32, 32, 3))
```

input



گام 2: تعریف node ها و اتصال به یکدیگر



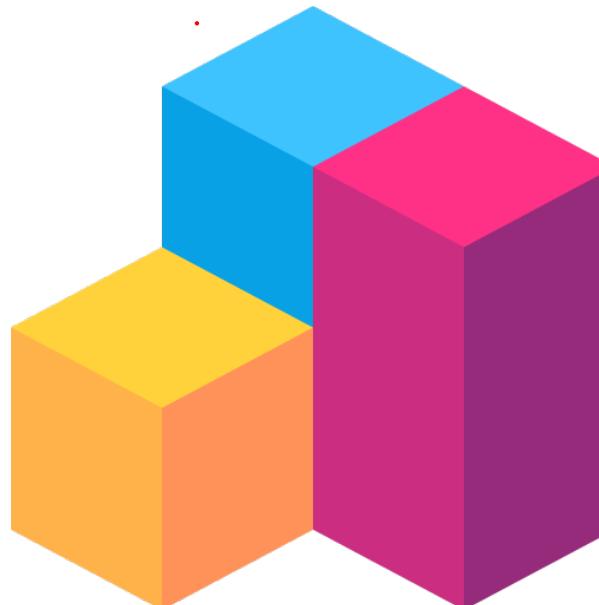
گام 3 : ساخت مدل با تعریف ورودی ها و خروجی ها

```
net = models.Model(inputs=inputs, outputs=outputs, name="mnist_model")
```

نور از ورودی

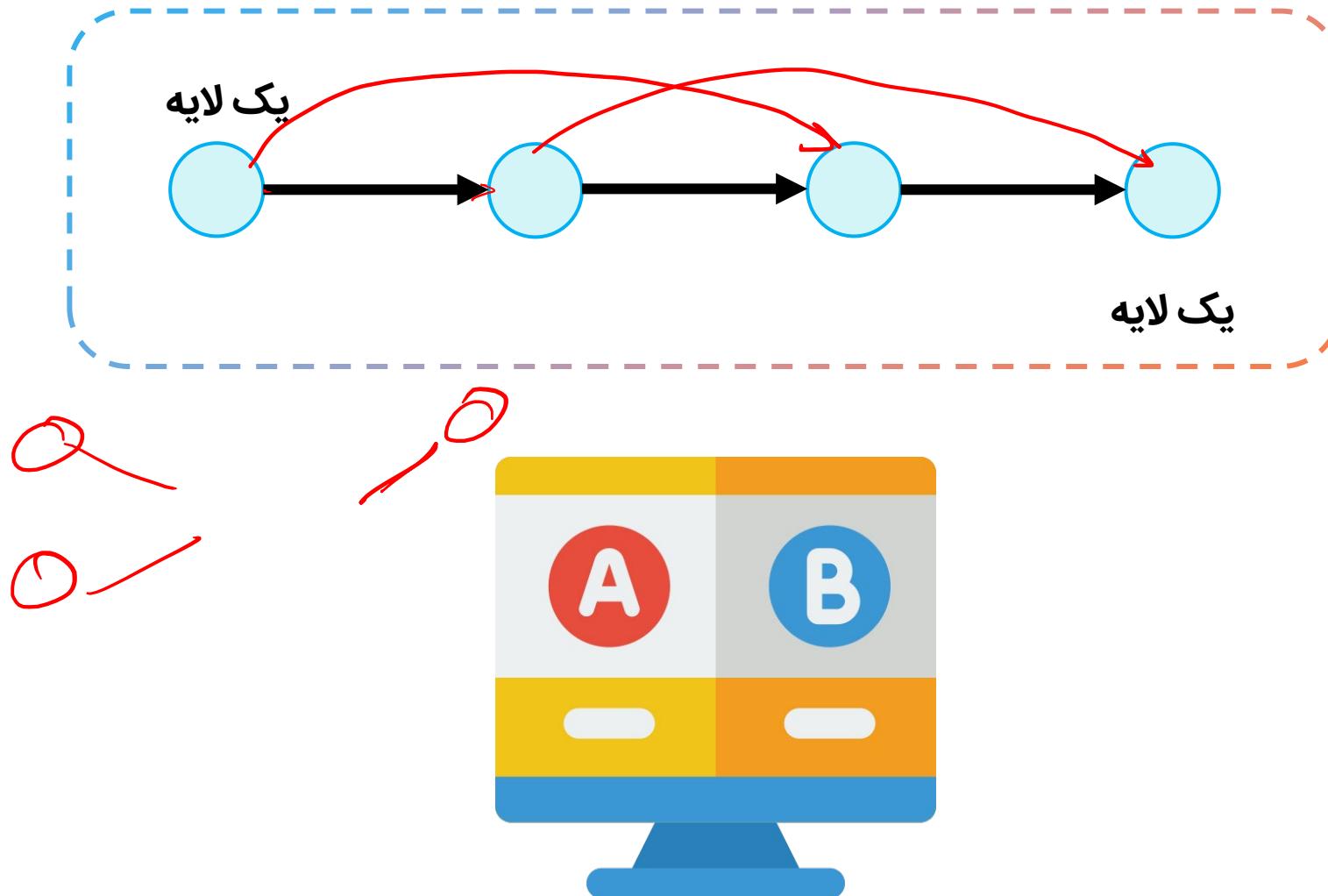
نور از خروجی

نام

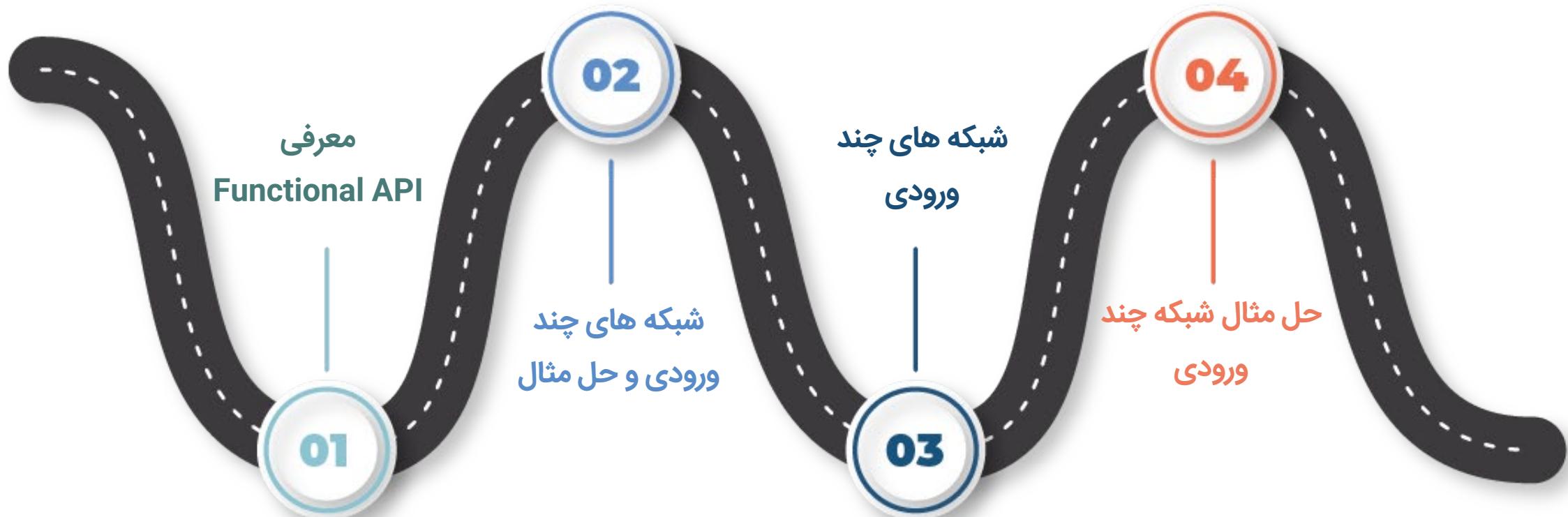


Connect -

مقایسه Sequential API با Functional API

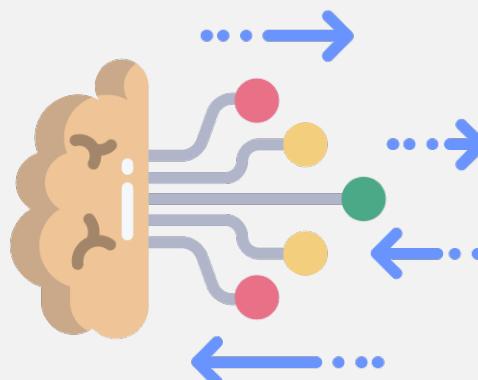


کجا هستیم؟



امروز دو مثال حل خواهیم کرد :

شبکه های چند خروجی

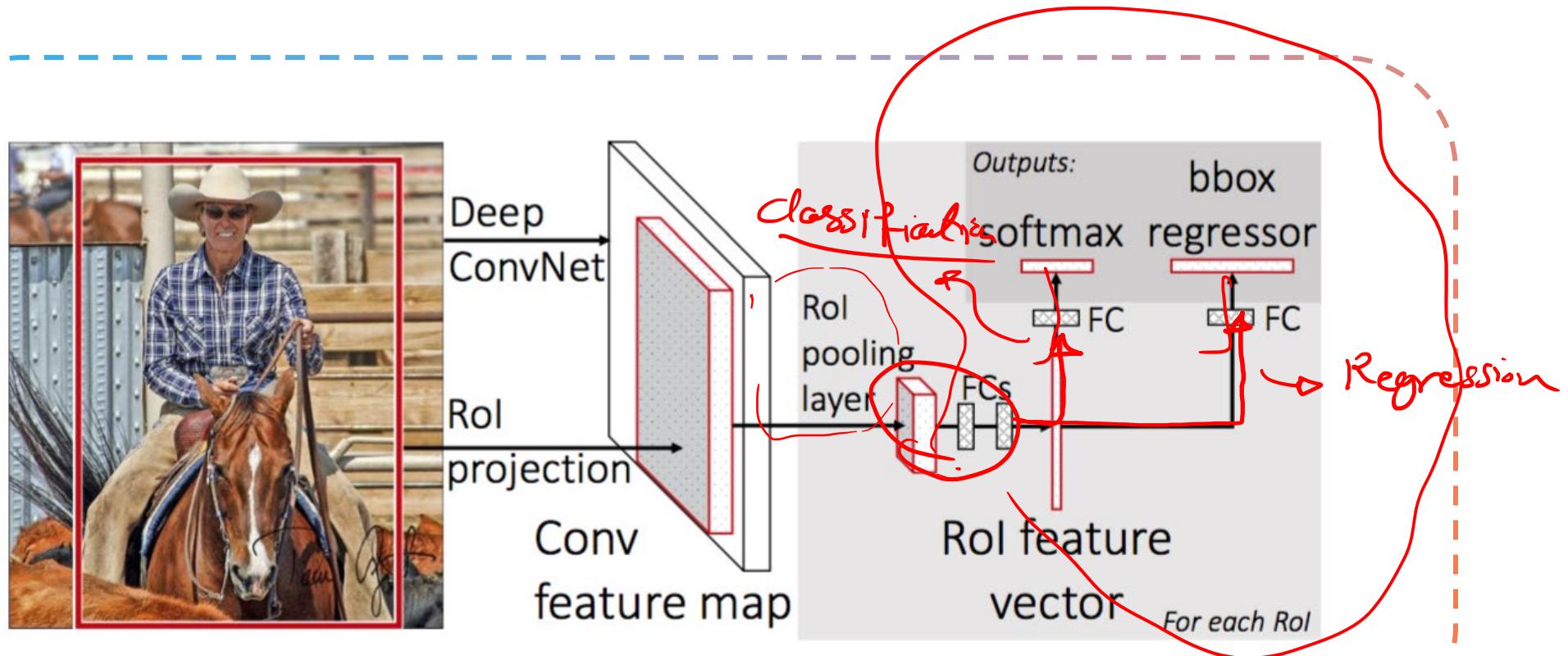


شبکه های چند ورودی



مثالی از شبکه های چند خروجی:

وخت
خت
چند خروجی
چند حرزل

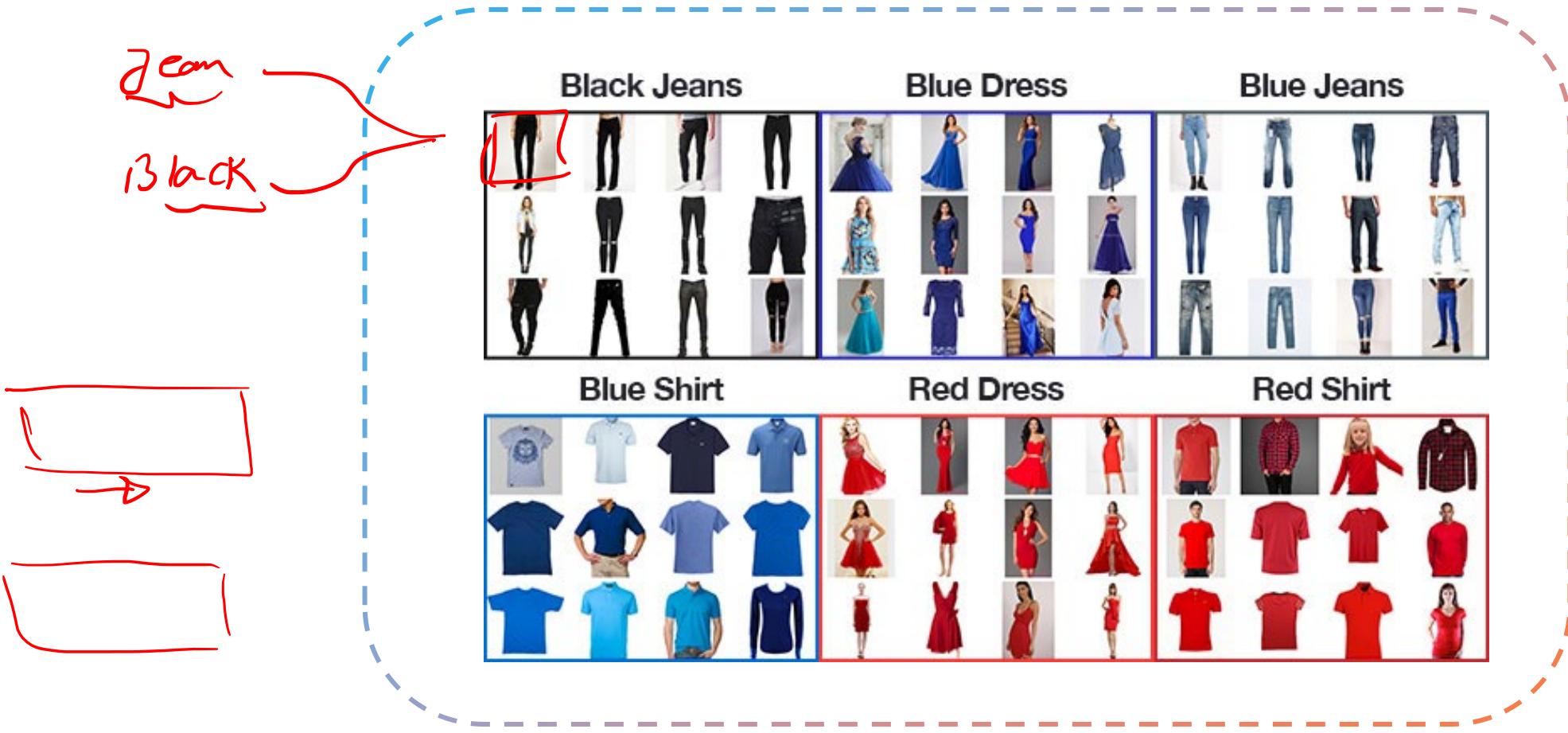


[GAN] $\xrightarrow{\text{پس از}} \xrightarrow{\text{معنی}} \xrightarrow{\text{معنی}}$

Fast RCNN - 2015

(object Detection)

مثالی از شبکه های چند خروجی:



Multi Label Classification

Multi Label Classification



ایده اصلی چیست؟

معکوس (ورودی‌های)



out more BN --

Car Man

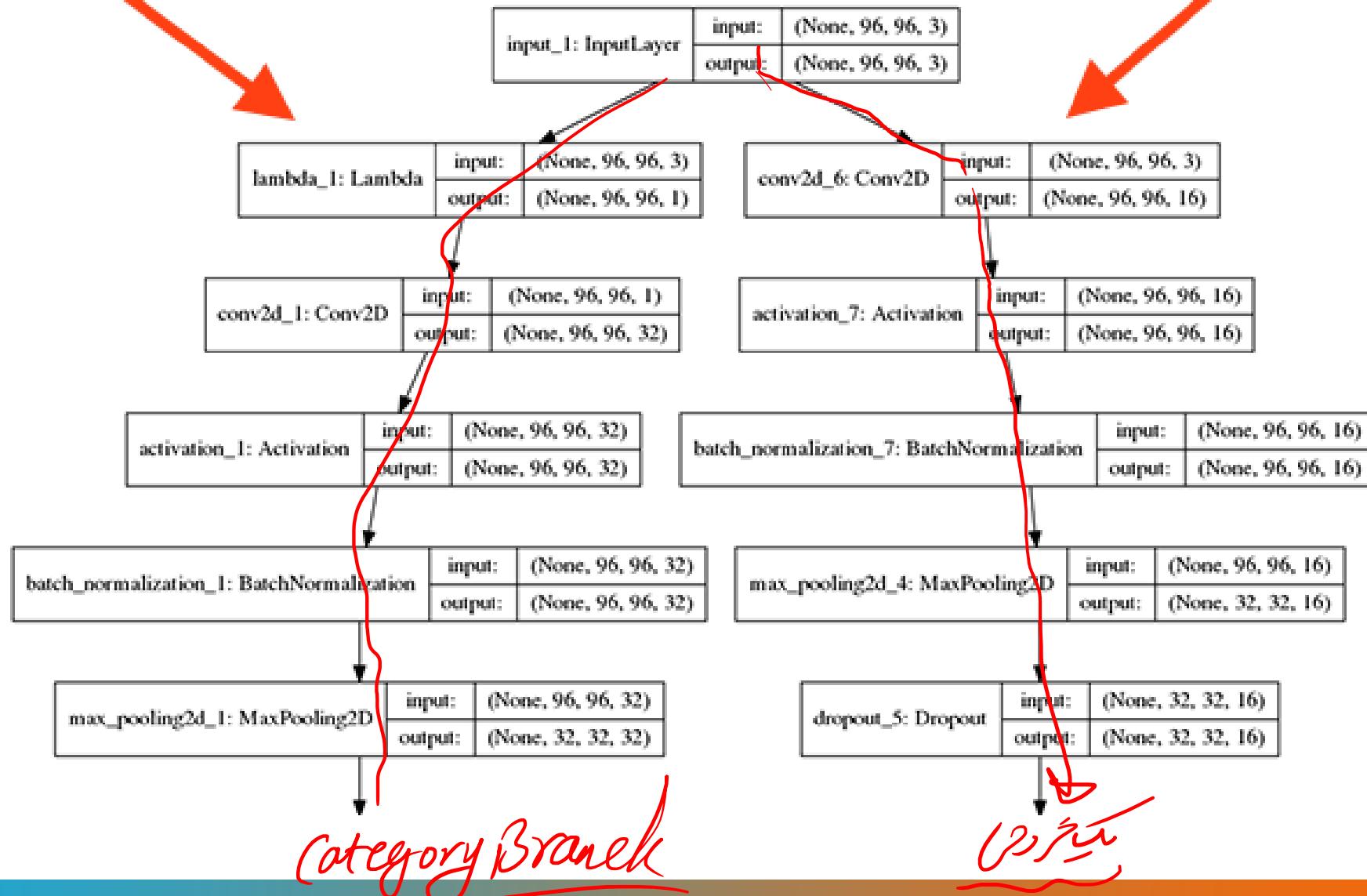
زگ آپیز فرنز کر
رو

Category

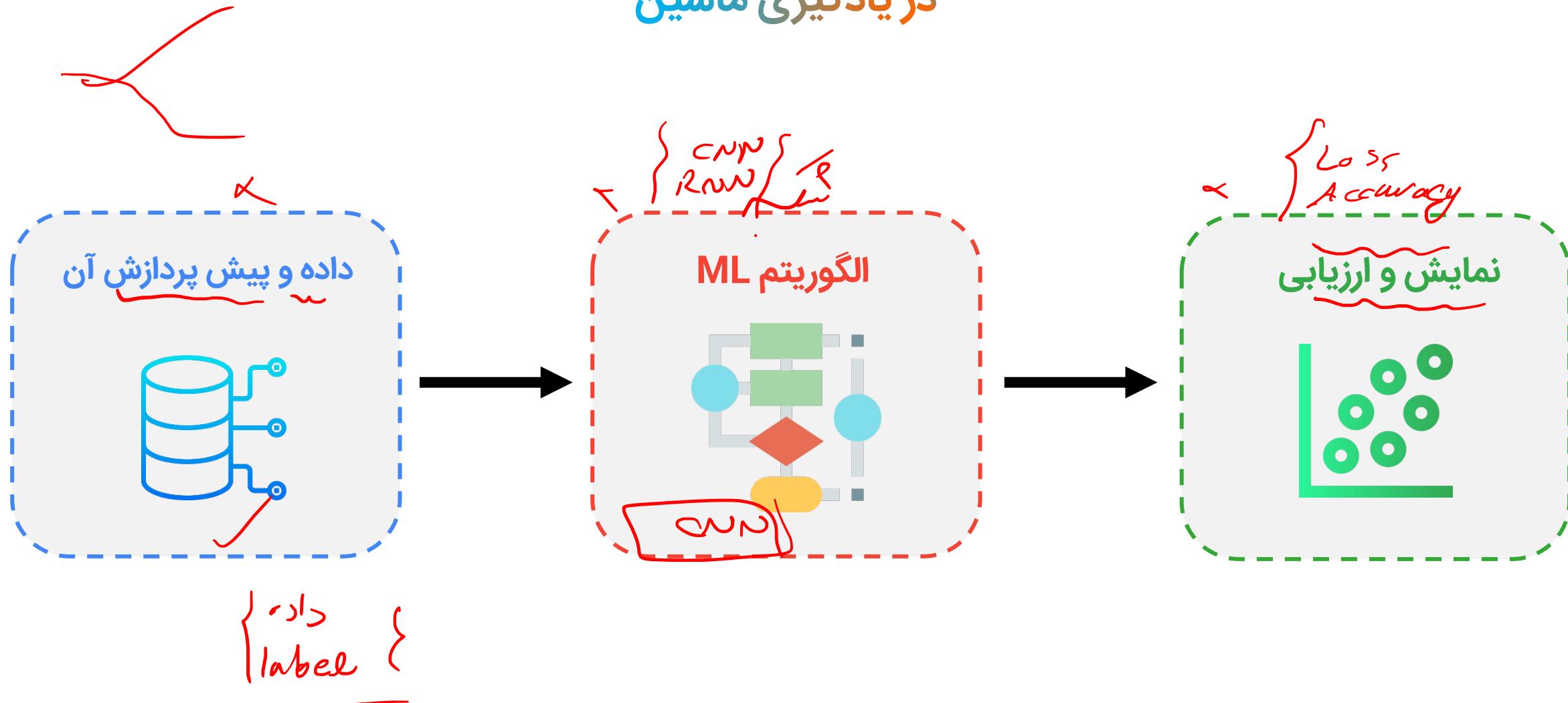
Category branch

Input image

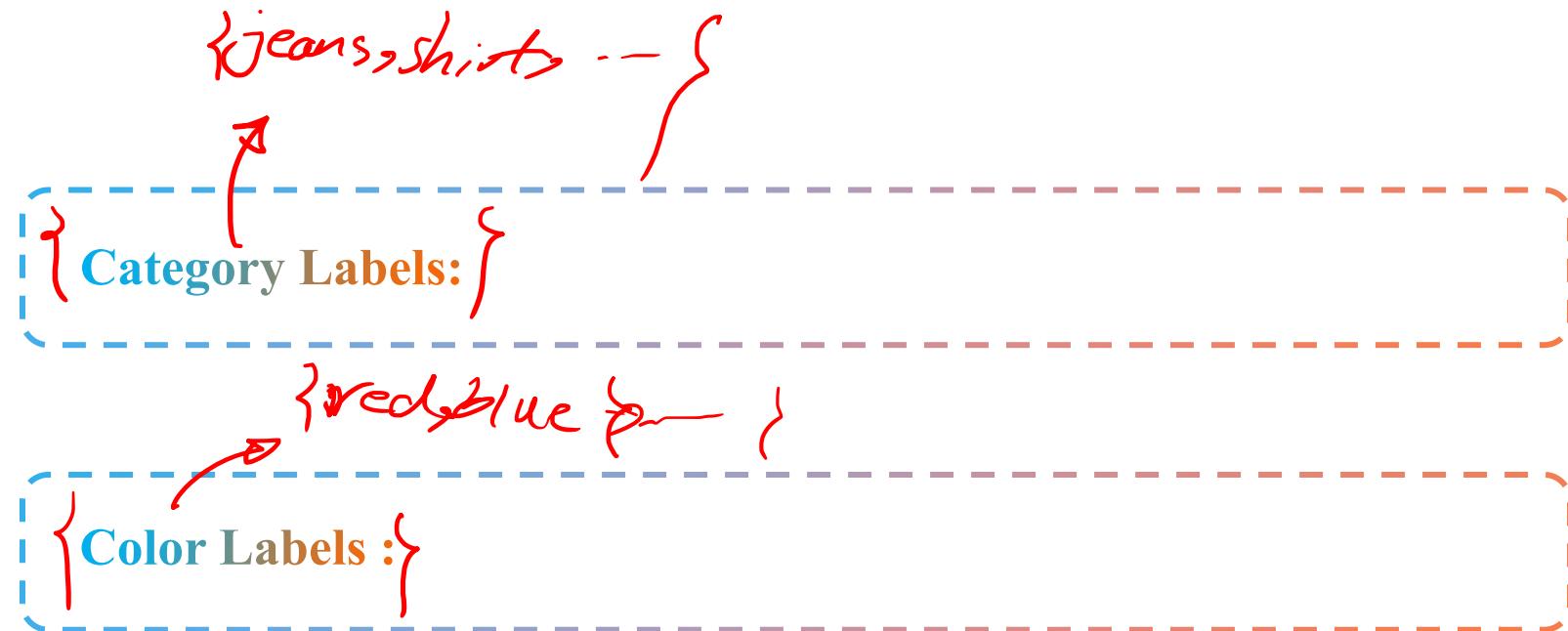
Color branch



در یادگیری ماشین

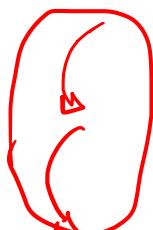


در بخش داده چه تغییری به وجود می آید ؟



هر کدام از Label ها باید جداگانه One Hot شوند.

sklearn



```
categoryLB = LabelBinarizer()  
colorLB = LabelBinarizer()  
category_labels = categoryLB.fit_transform(category_labels)  
color_labels = colorLB.fit_transform(color_labels)
```

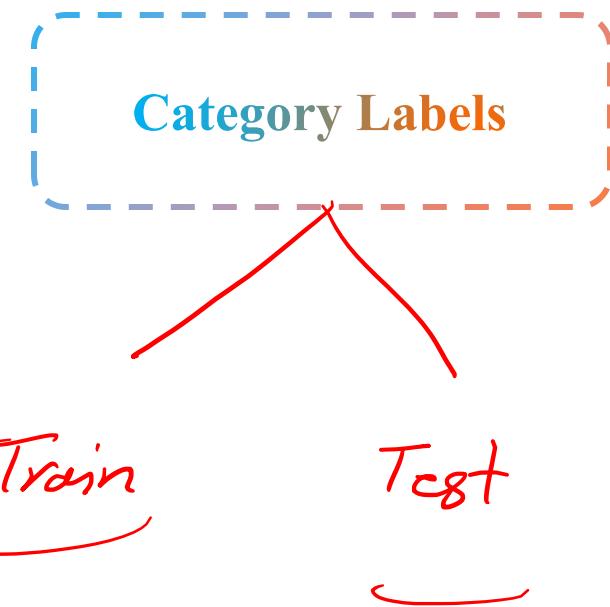
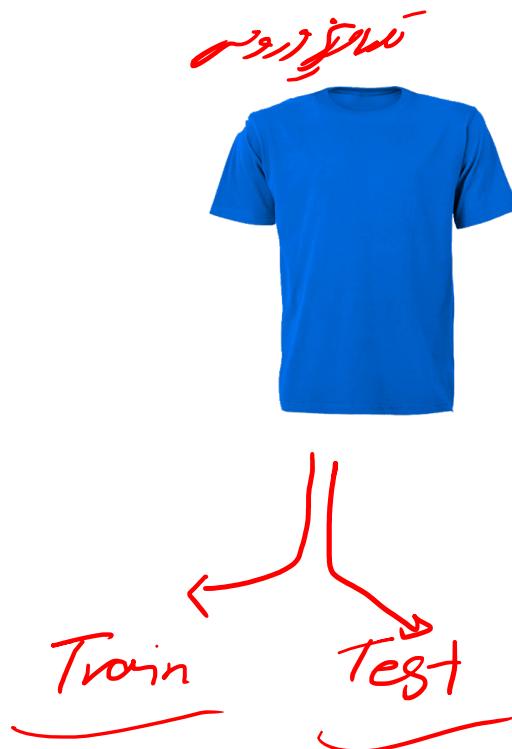
categoryLB = []

colorLB = []

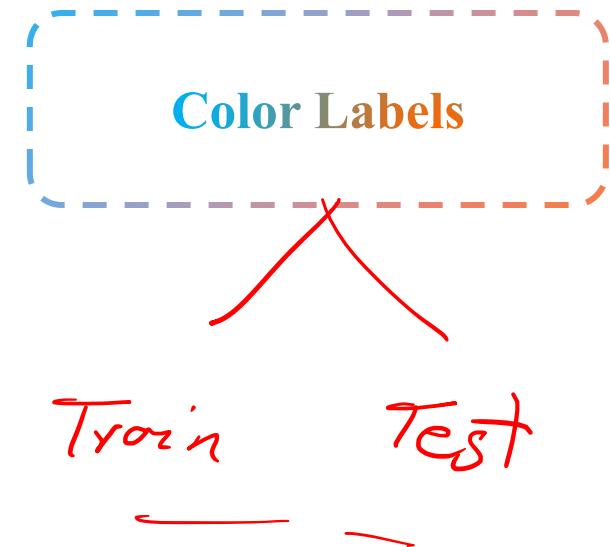
fit → ۱/۰

Transform ساخت

در قسمت چطور؟ Train Test Split

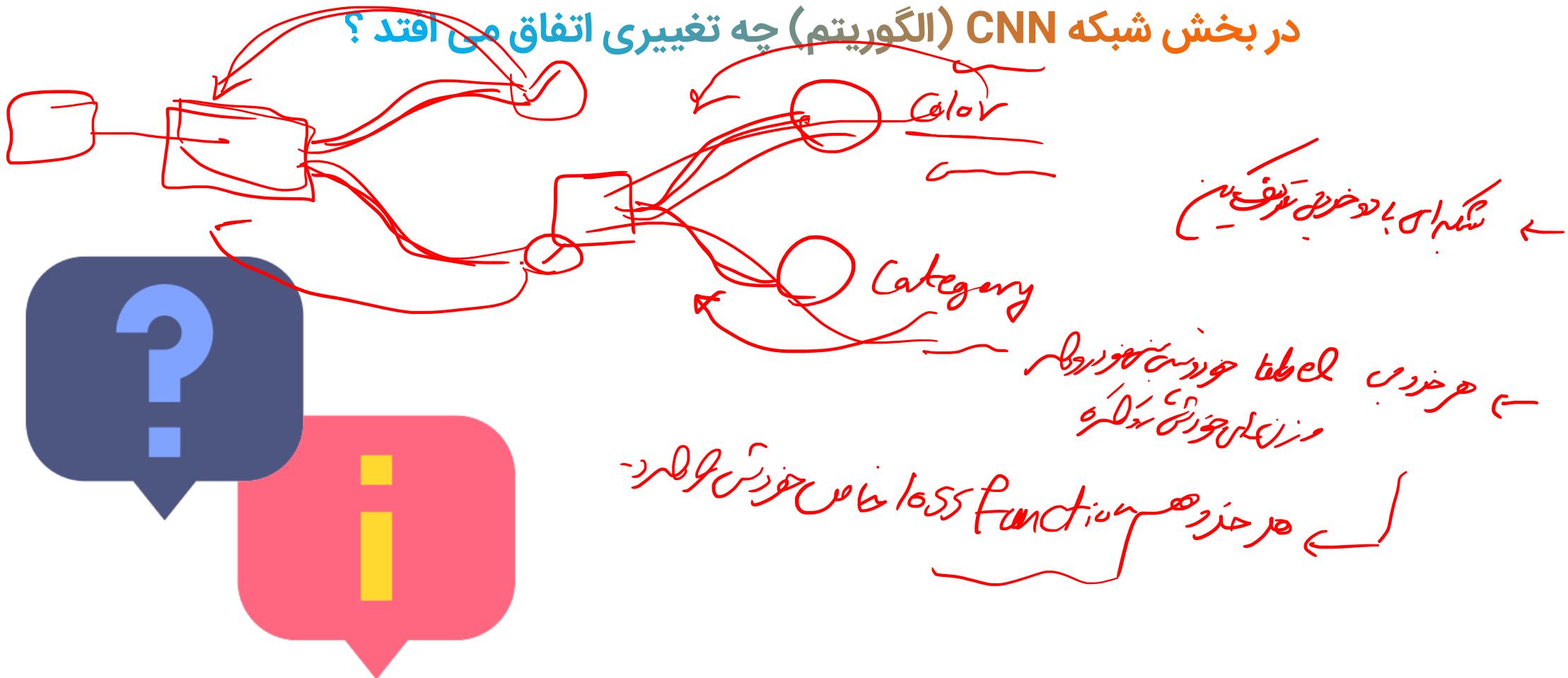


(X_{train}, X_{test})

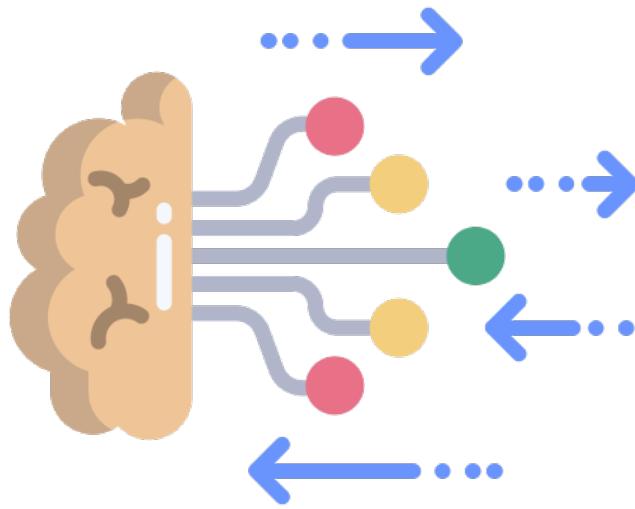


همین موضوع در کد

```
split = train_test_split(all_images, category_labels,  
                        color_labels, test_size=0.2)  
  
(trainX, testX, trainCategoryY, testCategoryY, trainColorY, testColorY) = split  
#(که) ۰.۲ >
```



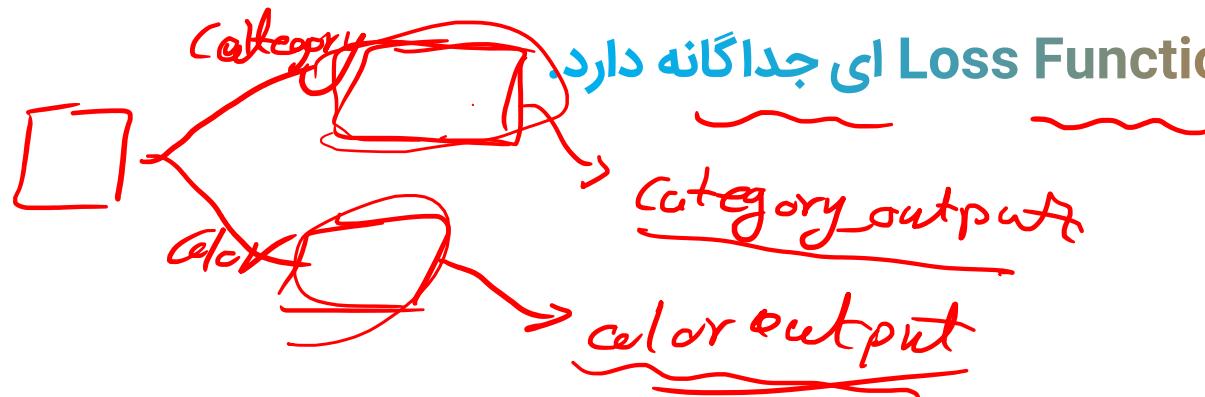
تغییر ۱: شبکه چند خروجی جداگانه خواهد داشت.



```
net = models.Model(inputs = input_layer,  
outputs = [cat_net, col_net],  
name = "fashionNet")
```

category خروجی
color خروجی

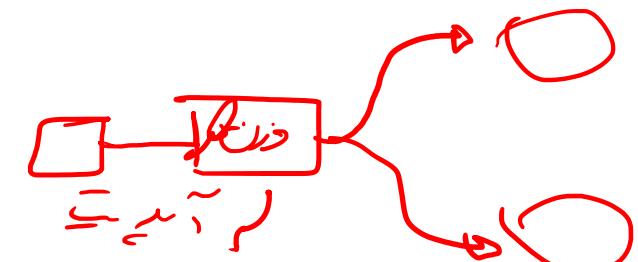
تغییر 2: هر خروجی Loss Function ای جداگانه دارد.



```
losses = {  
    "category_output": "categorical_crossentropy",  
    "color_output": "categorical_crossentropy",  
}
```

compil

[]



در پرانتز: می توانیم برای هر Loss یک ضریب هم در نظر بگیریم.

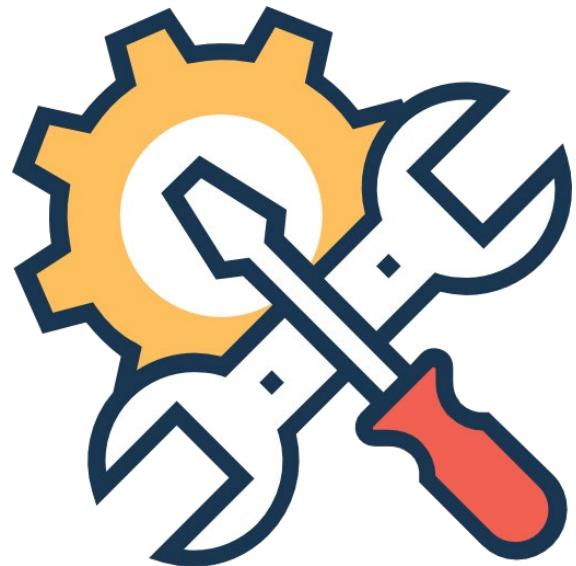
```
loss_weights = {"category_output": 1.0, "color_output": 1.0}
```

Loss1 + Loss2

وزن



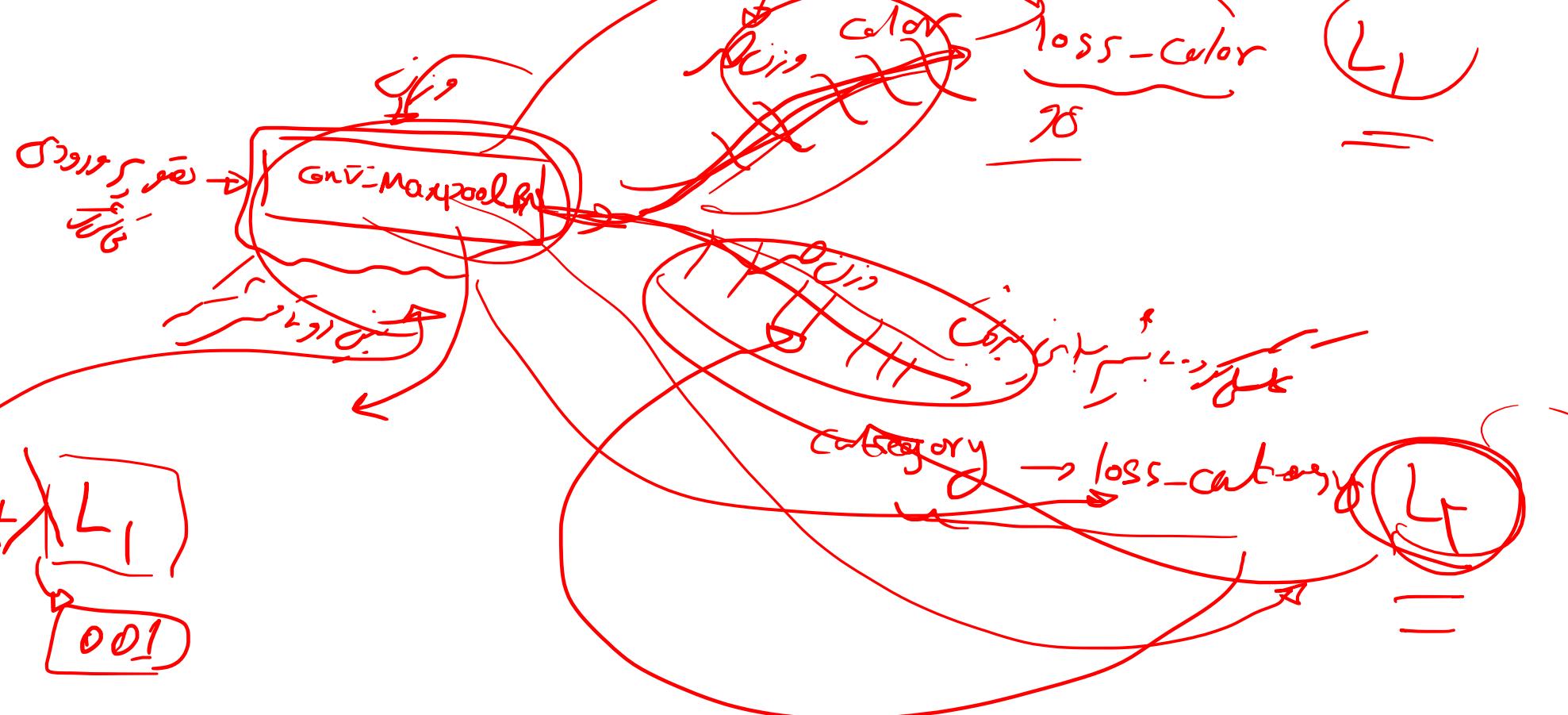
بنابراین `compile` مدل یک چنین شکلی خواهد داشت:



```
net.compile(optimizer="adam",
             loss = losses,
             loss_weights = loss_weights,
             metrics = ["accuracy"]))
```

classification → cat

$$L_I + L_F$$



خود ری
خود ری

در هنگام Train (استفاده از fit خروجی هر بخش مشخص شود).

```
H = net.fit(x=trainX,  
            y={"category_output": trainCategoryY, "color_output": trainColorY},  
            validation_data=(testX,  
                            {"category_output": testCategoryY, "color_output": testColorY}),  
            epochs=40, verbose = 1)
```

Train X
L1 Label trainCategory
L2 Label separate
Train color



label $\leftarrow L_1$
برچسب هایی که در مجموعه آموزشی
نمایند
 \rightarrow fit
C1

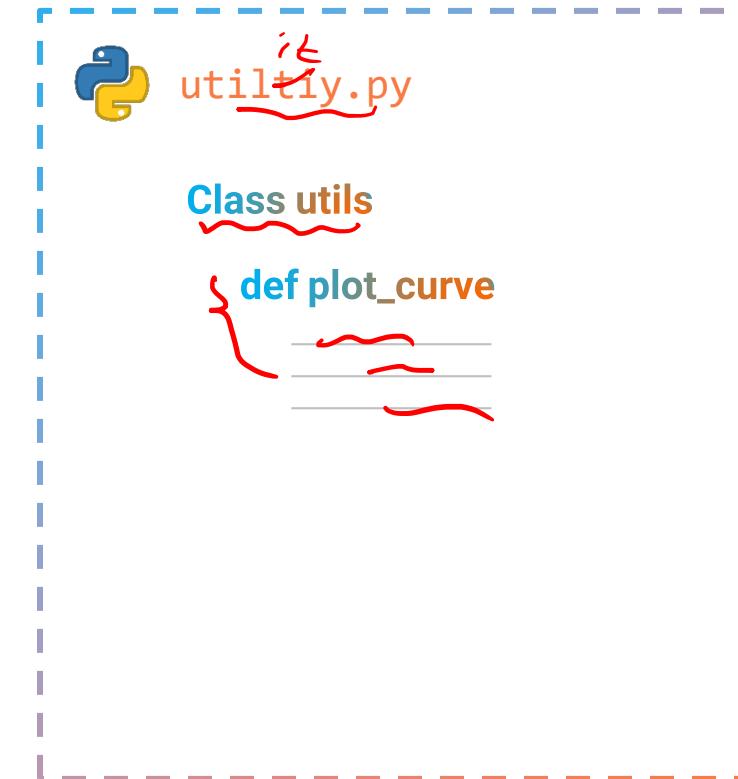
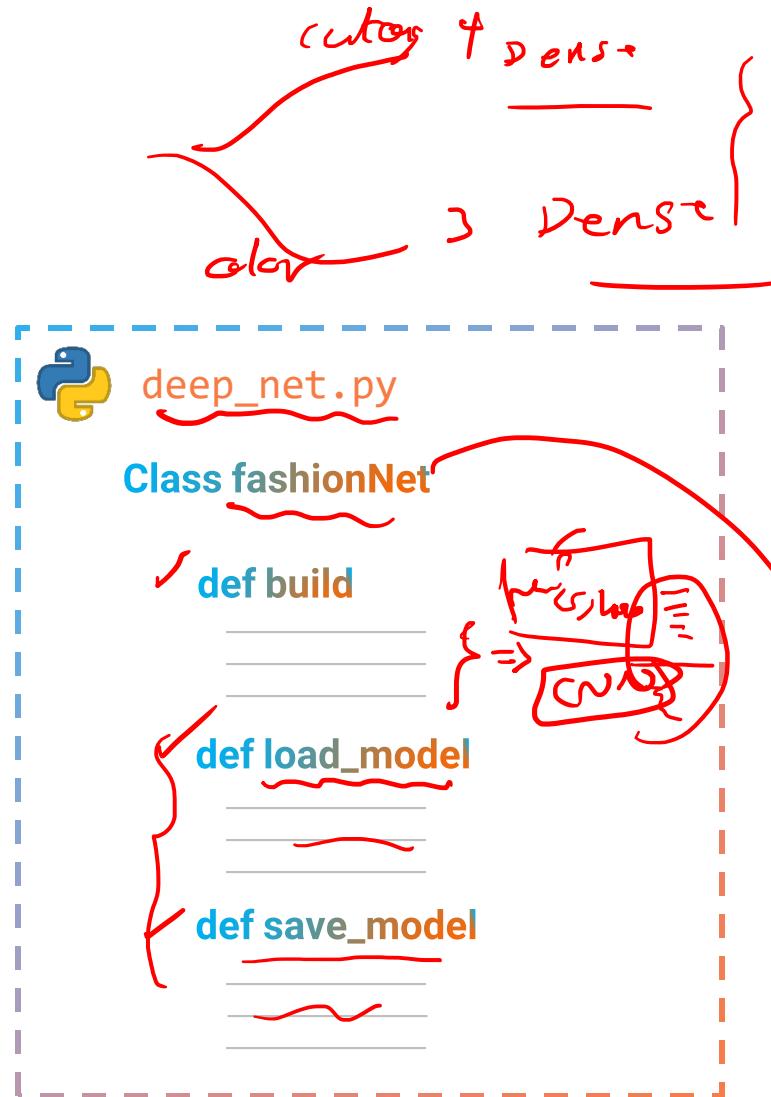
در بخش نمایش (رسم نمودار ها) چه تغییری اتفاق می افتد ؟

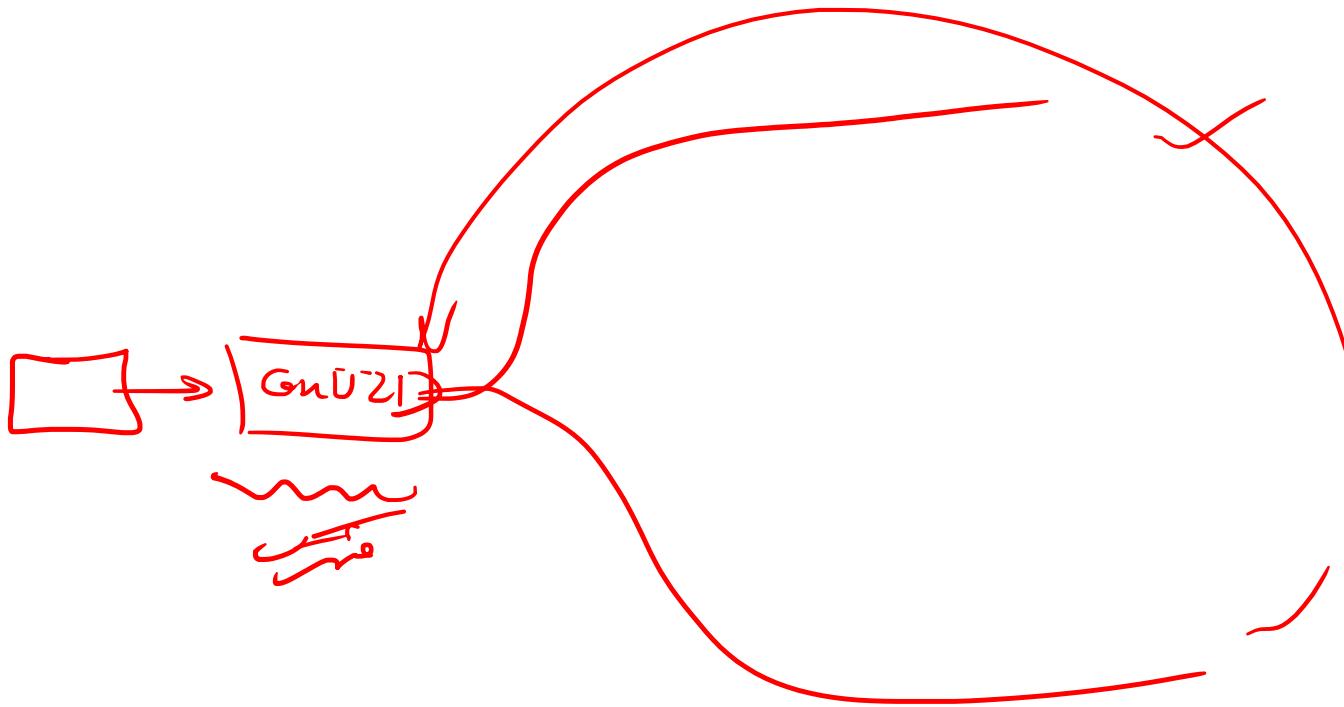
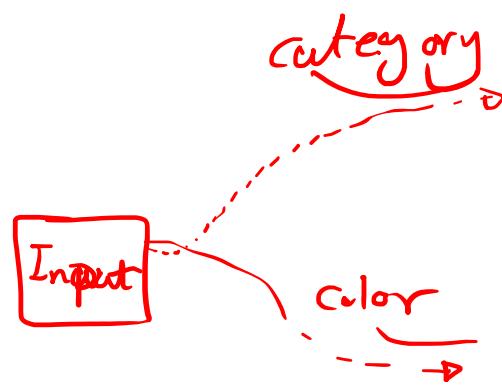
category_output_accuracy
اصغری

```
plt.plot(H.history['category_output_accuracy'], label = "category acc")
plt.plot(H.history["val_category_output_accuracy"], label="val category acc")
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.legend()
plt.show()
plt.close()
```

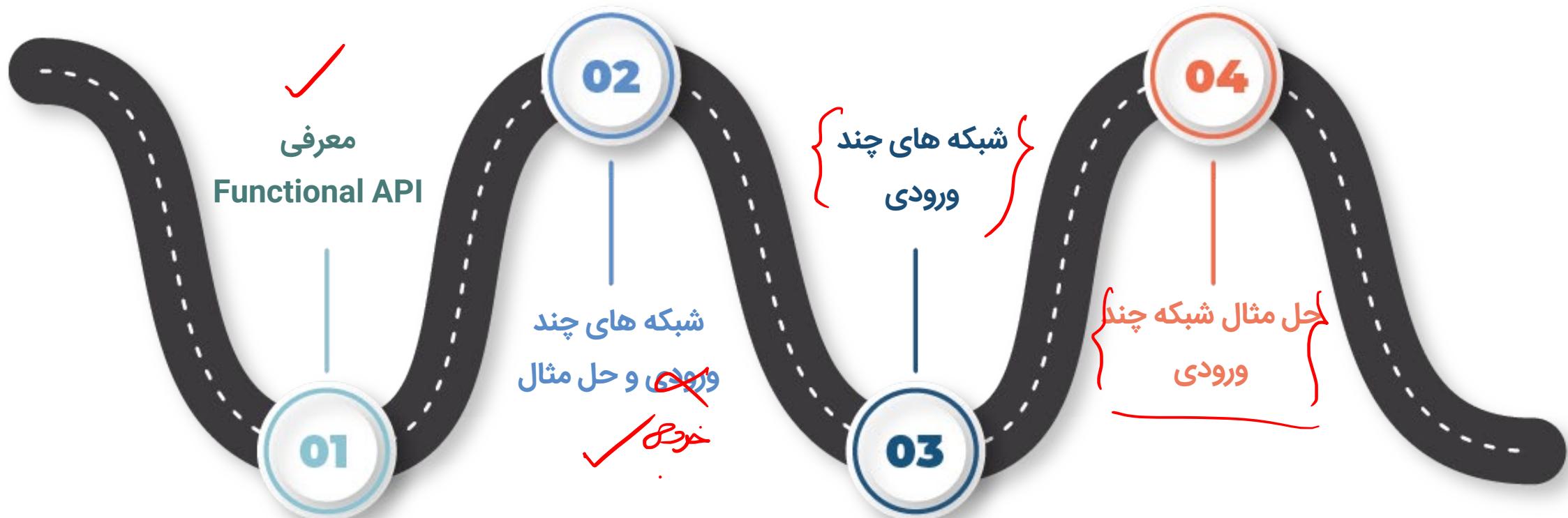


ساختار پروژه تعریف شده:

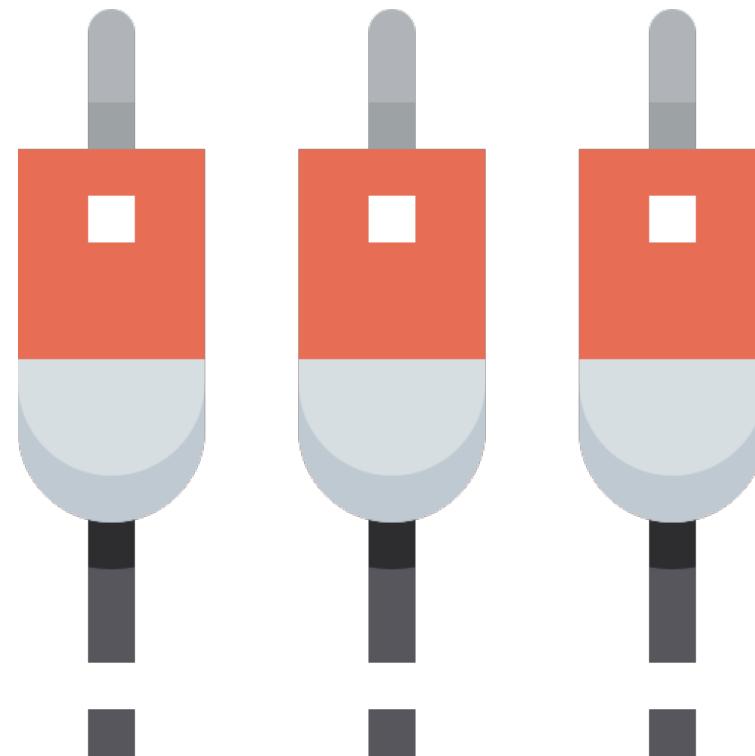




کجا هستیم؟



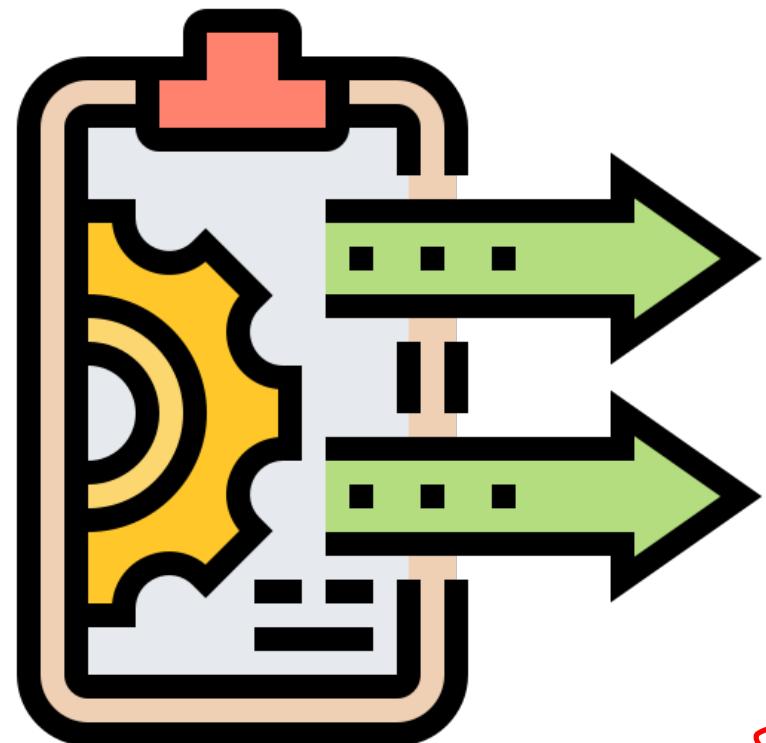
شبکه هایی با چند ورودی



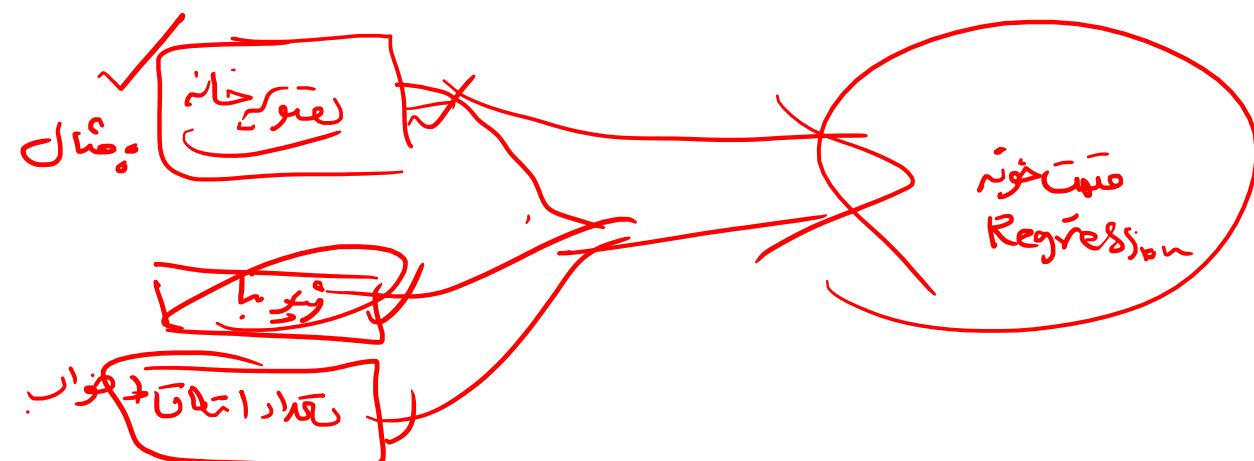
در یادگیری ماشین

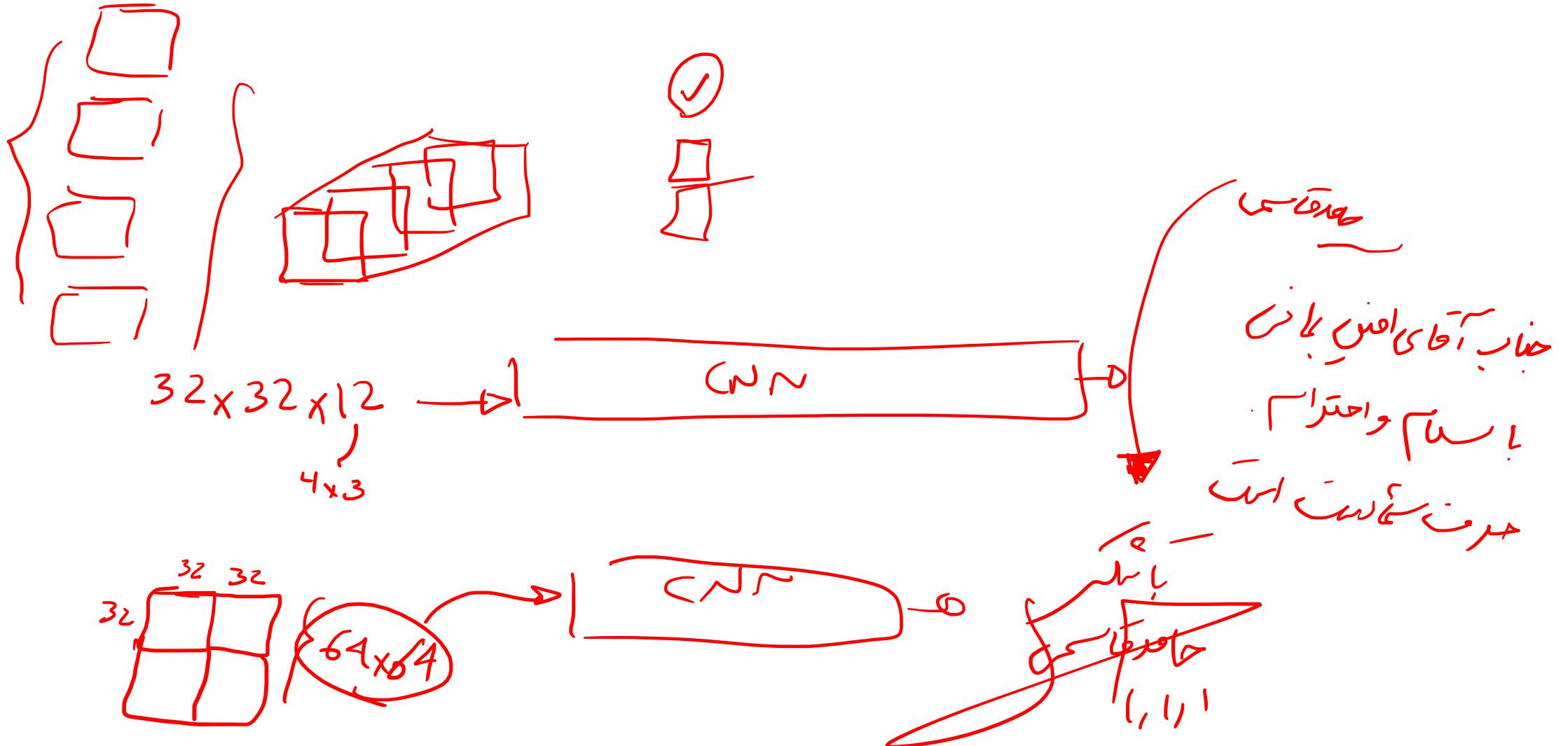


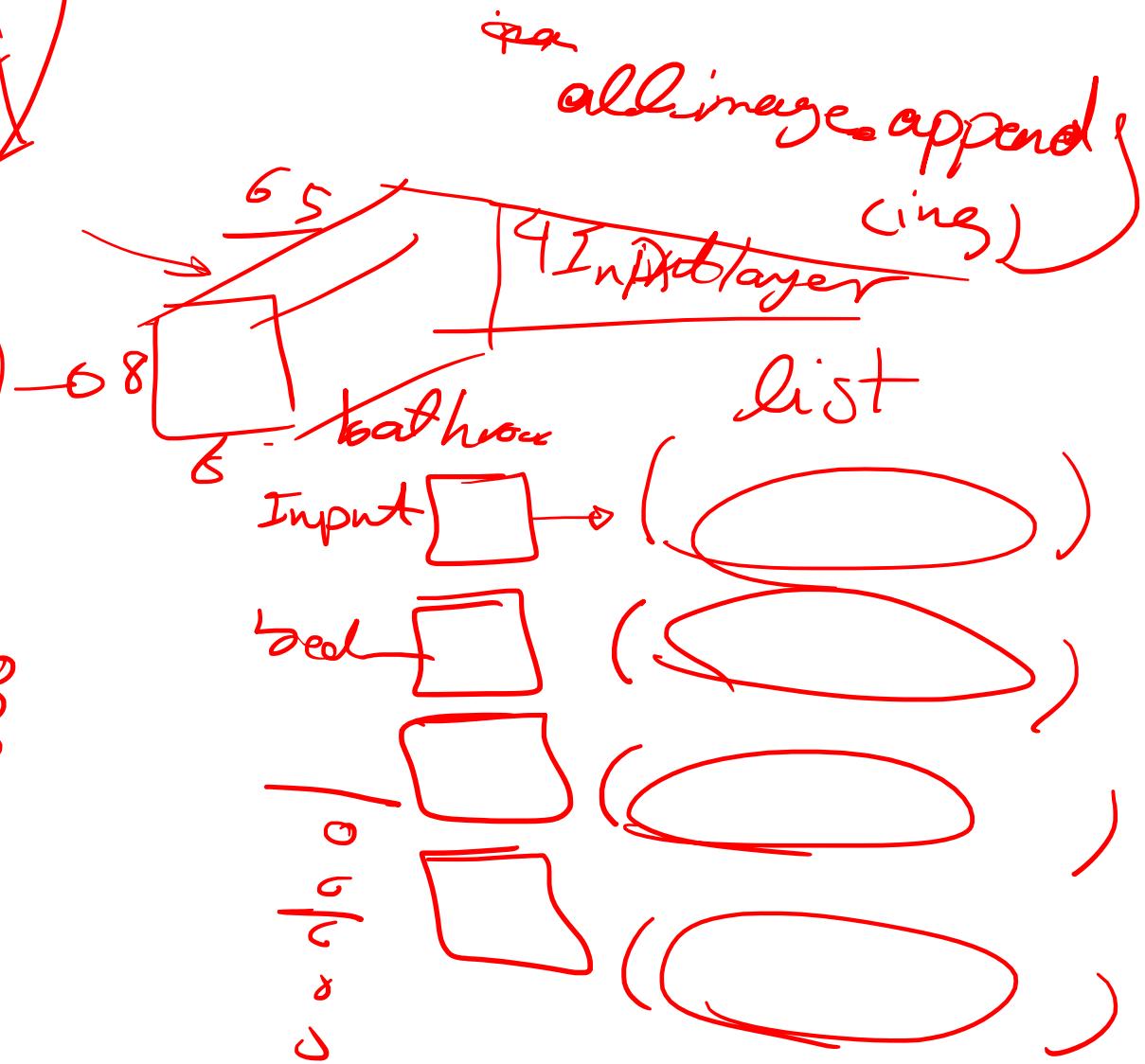
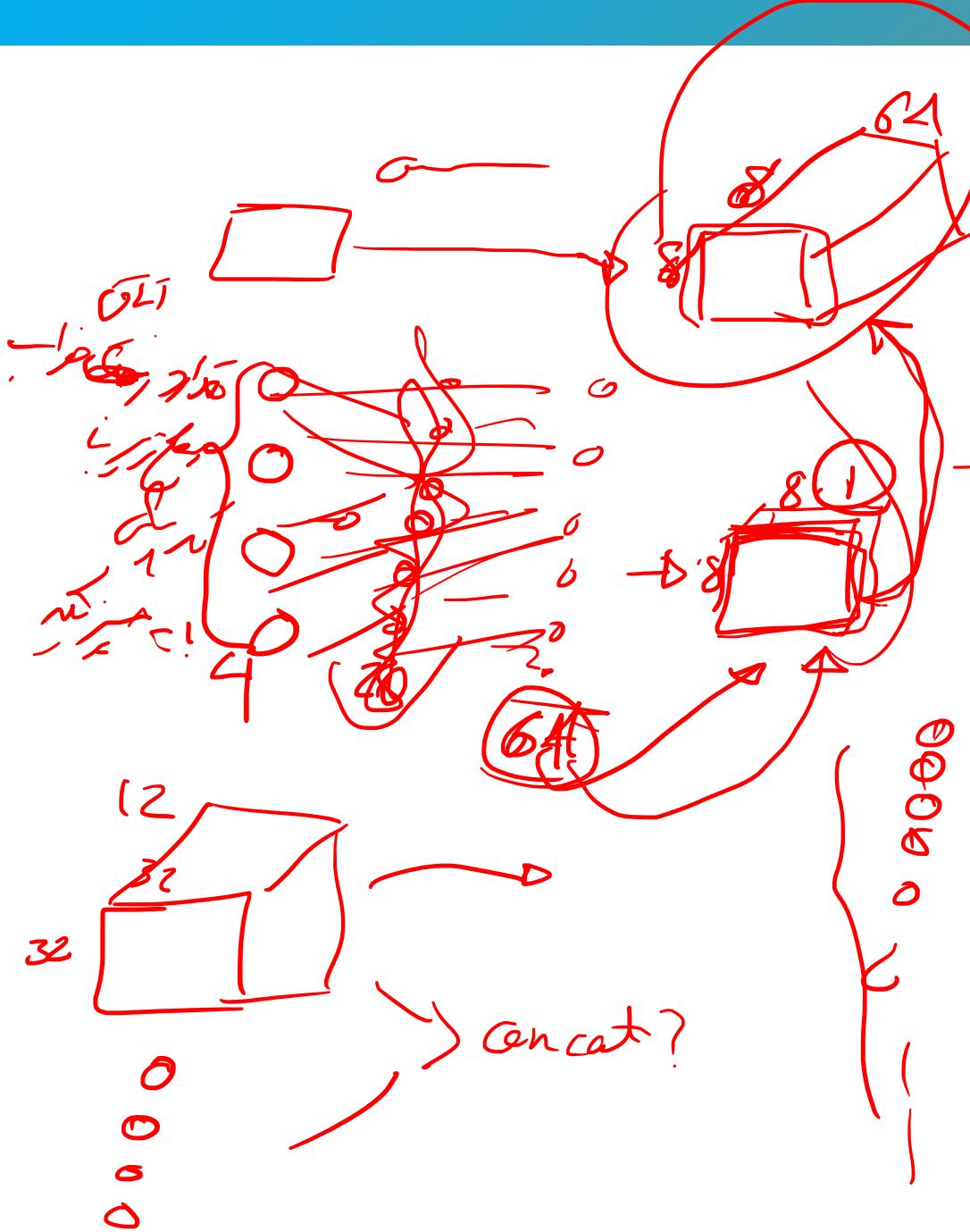
در بخش داده چه تغییراتی اتفاق می افتد ؟



هر ورودی لیست جداگانه داده دارد.

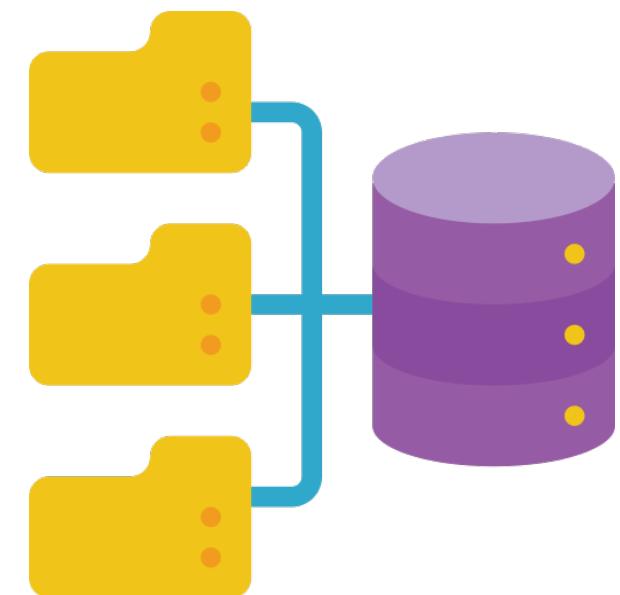






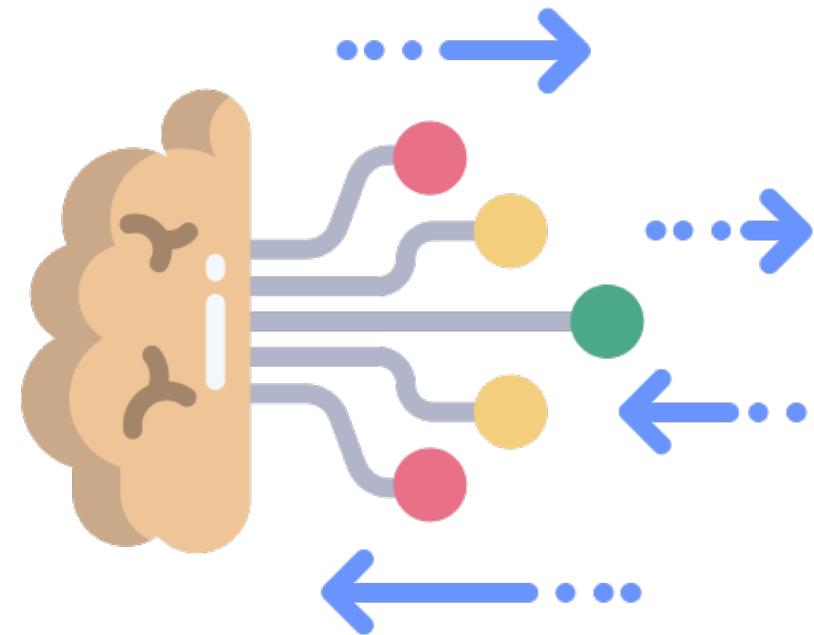
در بخش داده چه تغییراتی اتفاق می افتد ؟

```
for i, item in enumerate(glob.glob("house_dataset\\*.jpg")):  
    image = cv2.imread(item)  
    image = preprocess(image)  
    location = item.split("\\")[-1].split("_")[-1].split(".")[0]  
    if location == "bathroom": bathroom_list.append(image)  
    elif location == "bedroom": bedroom_list.append(image)  
    elif location == "frontal": frontal_list.append(image)  
    elif location == "kitchen": kitchen_list.append(image)
```



در بخش `cnn` (الگوریتم) چه اتفاقاتی می‌افتد؟

هر ورودی : `layers.Input` جداگانه خود را دارد.

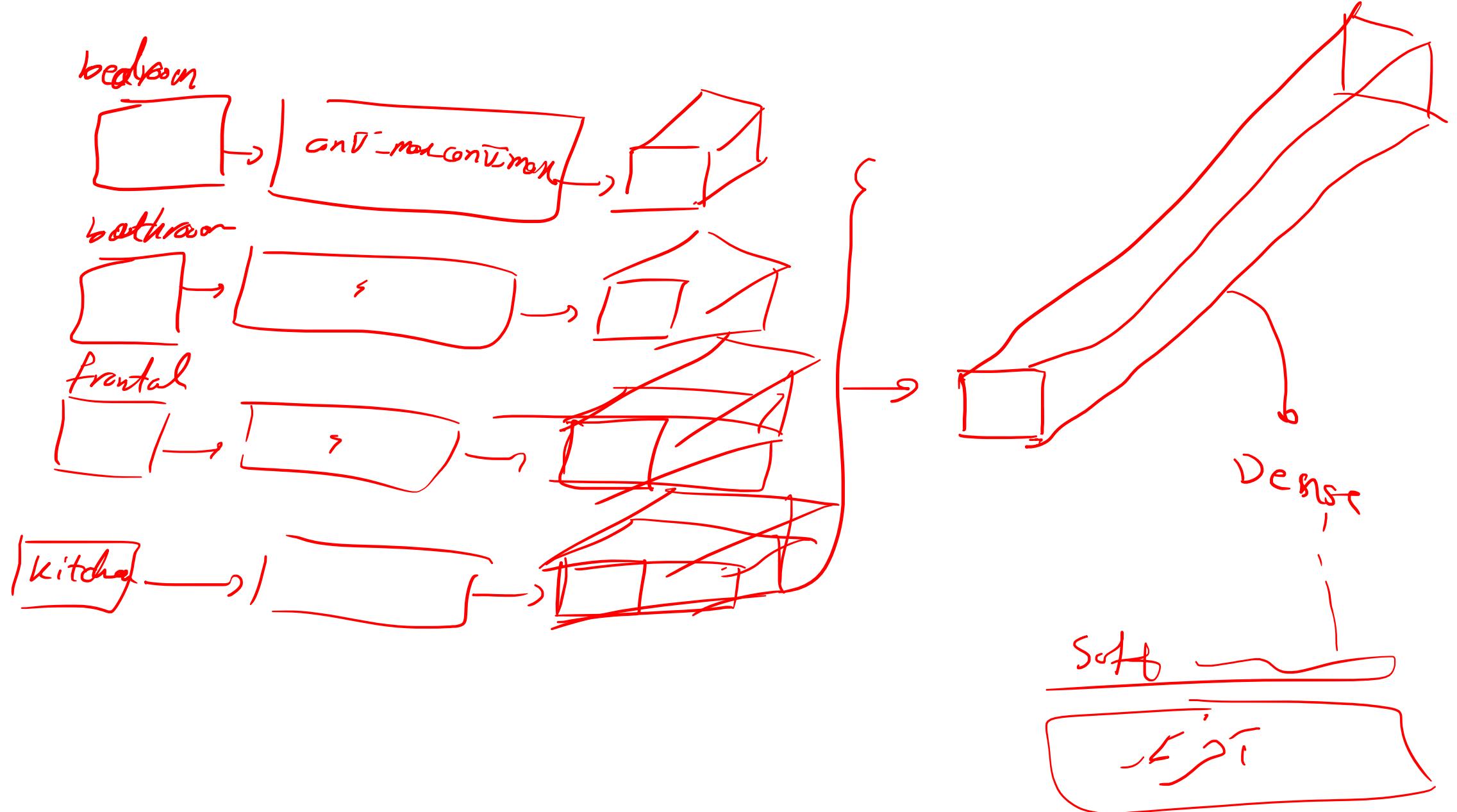


شبکه عصبی پیشنهاد شده:

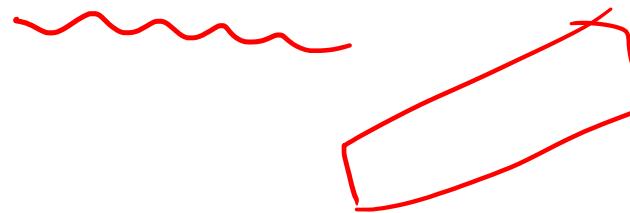
bathroom

```
bedroom_input = layers.Input((32, 32, 3))
x = layers.Conv2D(16, (3, 3), padding="same",
                  activation="relu")(bedroom_input)
x = layers.MaxPool2D((2, 2))(x)
x = layers.Conv2D(32, (3, 3), padding="same", activation="relu")(x)
x = layers.MaxPool2D((2, 2))(x)
```





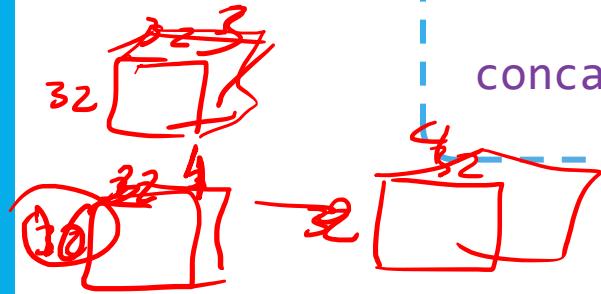
در نهایت ورودی ها در جایی باید به هم برسند !



```
concat_inputs = layers.concatenate([x, y, z, w], axis = 2)
flat_layer = layers.Flatten()(concat_inputs)
out = layers.Dense(100, activation="relu")(flat_layer)
out = layers.Dense(1, activation="linear")(out)
```

regress.

layers.concatenate معرفی



```
concat_inputs = layers.concatenate([x, y, z, w], axis = 2)
```

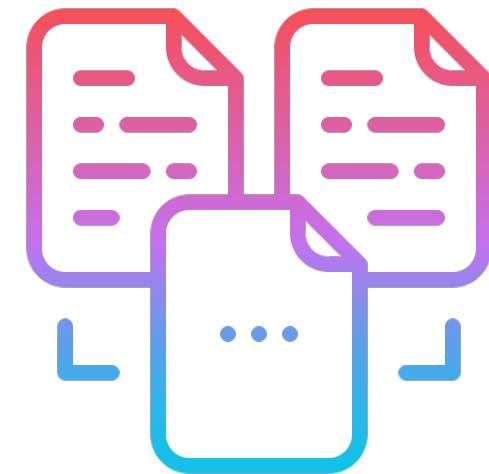
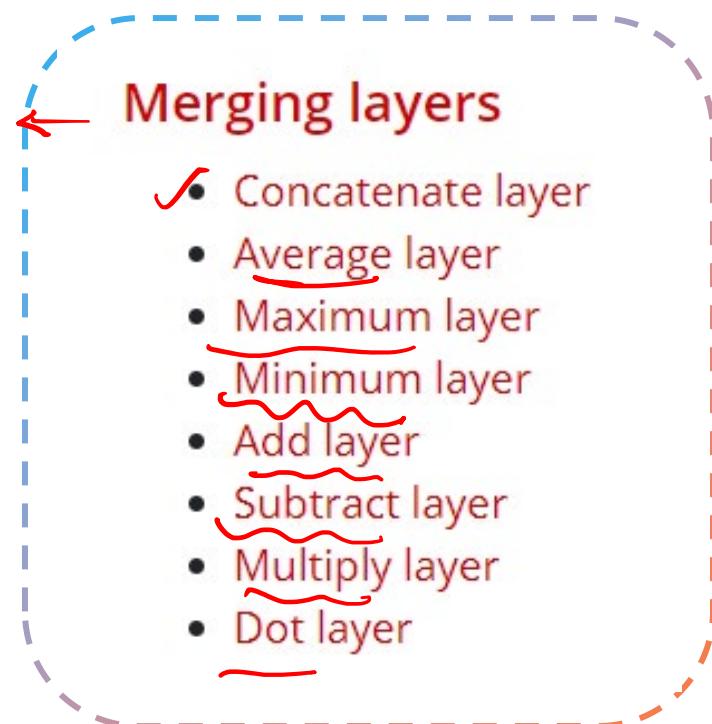
✓ دریافت لیستی از ورودی ها به صورت تنسور

✓ ابعاد ورودی ها باید یکی باشد به جز در راستای محور concat

✓ یکی از Merging Layer ها



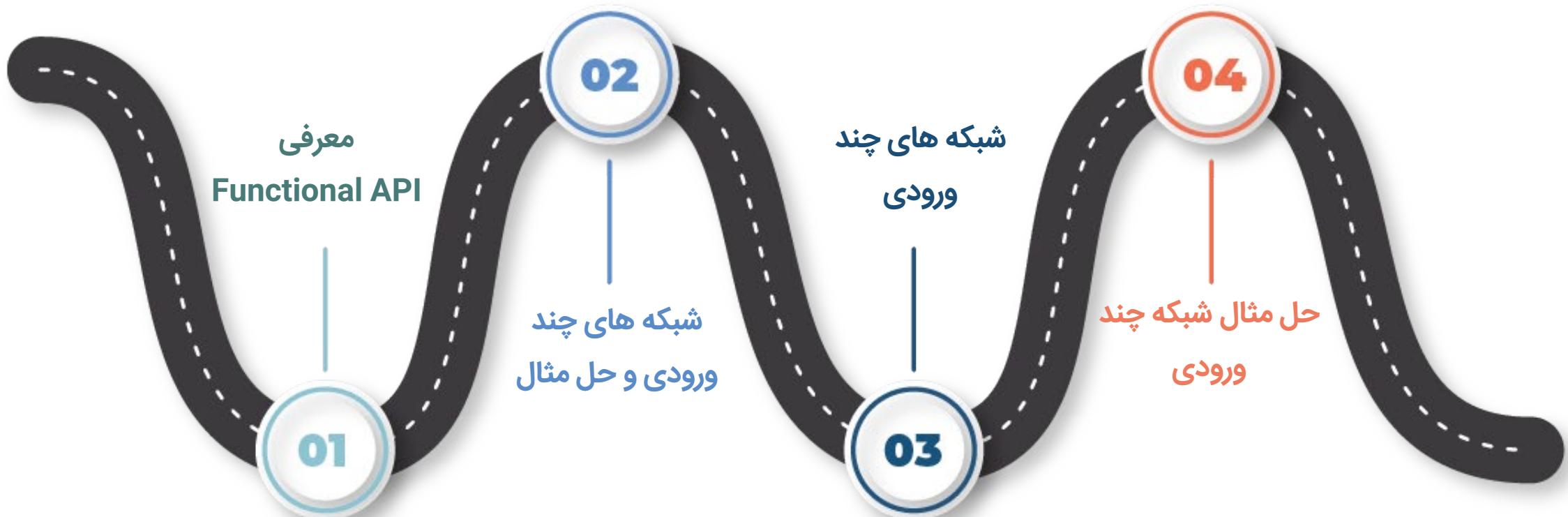
لایه های Merging برای ترکیب ورودی ها



منبع مورد استفاده



پایان



Special Method in Python

```
print(1+2)  
print("a" + "b")
```

```
human_1 = Human(178, 71, "hamed", "ghasemi")  
print(human_1)
```



```
<__main__.Human object at 0x00000253C621DFD0>
```

متده __repr__()

```
def __repr__(self):
    return "Human({}, {}, {}, {})".format(self.h, self.w, self.first, self.last)

human_1 = Human(178, 71, "hamed", "ghasemi")

print(human_1)
```

متده _str_()

```
def __str__(self):  
    return "{} {}, w = {}, h = {}".format(self.first, self.last, self.h, self.w)
```

```
print(repr(human_1))  
print(str(human_1))  
  
print(human_1.__repr__())  
print(human_1.__str__())
```

متده __add__()

```
print(1 + 2)  
  
print(int.__add__(1, 2))  
print(str.__add__("a", "b"))
```

```
def __add__(self, other):  
    return self.w + other.w  
  
human_1 = Human(178, 71, "hamed", "ghasemi")  
human_2 = Human(182, 81, "hamed", "ghasemi")  
  
print(human_1 + human_2)
```

و البته عملگرهای دیگری نیز وجود دارد.

<i>client operation</i>	<i>special method</i>	<i>description</i>
$x + y$	<code>__add__(self, other)</code>	<i>sum of x and y</i>
$x - y$	<code>__sub__(self, other)</code>	<i>difference of x and y</i>
$x * y$	<code>__mul__(self, other)</code>	<i>product of x and y</i>
$x ** y$	<code>__pow__(self, other)</code>	<i>x to the yth power</i>
x / y	<code>__truediv__(self, other)</code>	<i>quotient of x and y</i>
$x // y$	<code>__floordiv__(self, other)</code>	<i>floored quotient of x and y</i>
$x \% y$	<code>__mod__(self, other)</code>	<i>remainder when dividing x by y</i>
$+x$	<code>__pos__(self)</code>	x
$-x$	<code>__neg__(self)</code>	<i>arithmetic negation of x</i>

Note: Python 2 uses `__div__` instead of `__truediv__`.

Special methods for arithmetic operators

len مت مت

```
def __len__(self):  
    return len(self.first) + len(self.last)  
  
human_1 = Human(178, 71, "hamed", "ghasemi")  
  
print(len(human_1))
```